INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

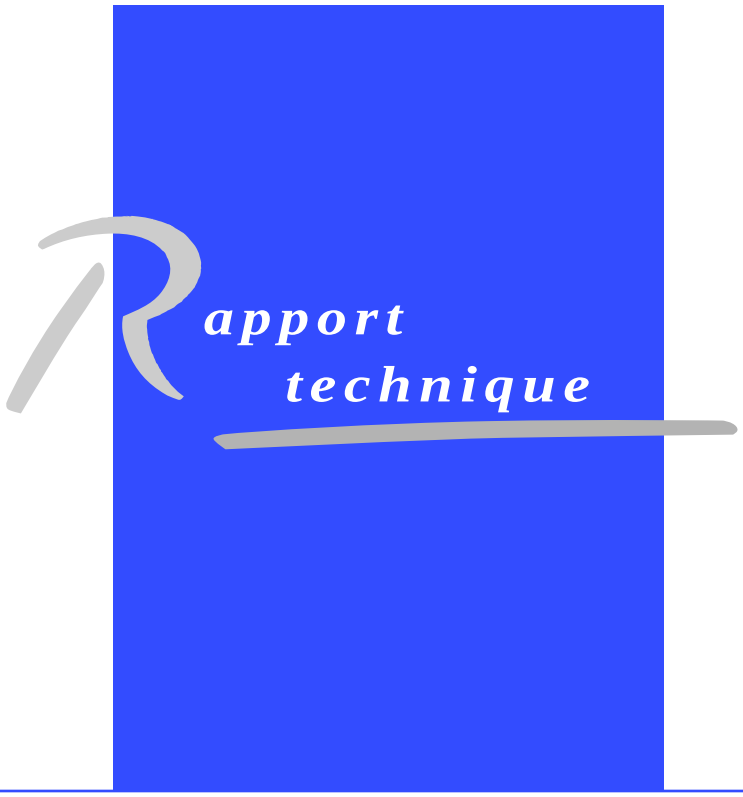
BLSURF – Mailleur de surfaces composées de carreaux paramétrés – Manuel d'utilisation

Patrick Laug , Houman Borouchaki

No 0232

4 juin 1999

THÈME 4



ISSN 0249-0803

BLSURF – Mailleur de surfaces composées de carreaux paramétrés – Manuel d'utilisation

Patrick Laug* , Houman Borouchaki**

Thème 4 — Simulation et optimisation
de systèmes complexes
Projet Gamma

Rapport technique n° 0232 — 4 juin 1999 — 48 pages

Résumé : Ce rapport technique décrit les fonctionnalités du mailleur automatique surfacique BLSURF. Ce logiciel génère le maillage d'une surface composée de plusieurs carreaux paramétrés, en respectant un champ de tailles spécifiées. À cette fin, les fonctions analytiques définissant la surface, ainsi que leurs dérivées premières et éventuellement secondes, devront être fournies de manière externe (par exemple par un système de CAO). En outre, chaque carreau sera décrit à partir de son domaine de paramètres 2D. La méthode utilisée est basée sur une approche incrémentale de Delaunay adaptée à une métrique riemannienne.

Mots-clé : maillage surfacique, surface composée paramétrée, métrique riemannienne, triangulation de Delaunay, approche frontale.

(Abstract: pto)

* INRIA, Institut National de Recherche en Informatique et en Automatique, UR de Rocquencourt, BP 105, 78153 Le Chesnay cedex, France. Email: Patrick.Laug@inria.fr.

** UTT, Université de Technologie de Troyes, 12 rue Marie Curie, BP 2060, 10010 Troyes cedex, France. Email: Houman.Borouchaki@univ-troyes.fr.

Unité de recherche INRIA Rocquencourt
Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex (France)
Téléphone : 01 39 63 55 11 - International : +33 1 39 63 55 11
Télécopie : (33) 01 39 63 53 30 - International : +33 1 39 63 53 30

BLSURF
Mesh Generator for Surfaces Composed of Parametric Patches
User's Manual

Abstract: This technical report describes the automatic surface mesh generator BLSURF. This software creates the mesh of a composite parametric surface, conforming to a prescribed size map. To this end, analytical functions defining the surface, and their first and possibly second derivatives, must be provided externally (for instance by a CAD system). Moreover, each patch should be described from its 2D parametric domain. The method used is based on an incremental Delaunay approach, adapted to a Riemannian metric.

Key-words: surface mesh, composite parametric surface, Riemannian metric, Delaunay triangulation, advancing front approach.

Table des matières

1 Introduction

Cette introduction présente brièvement le but et les fonctionnalités du logiciel BLSURF, puis quelques exemples viennent illustrer ses possibilités.

1.1 Présentation

Le maillage des surfaces tridimensionnelles joue un rôle très important dans de nombreux domaines du calcul scientifique, notamment la méthode des éléments finis. Il peut constituer une première étape pour la construction d'un maillage volumique. Une grande variété de surfaces peuvent être définies par des ensembles de *carreaux* paramétrés. Par exemple, les modeleurs de CAO définissent généralement les surfaces par des carreaux polynomiaux ou rationnels.

La méthode implémentée dans le logiciel BLSURF permet de générer un maillage qui respecte certaines contraintes (tailles souhaitées des éléments au voisinage des points de la surface) et qui approche fidèlement la "géométrie" de la surface. Elle consiste à mailler les domaines des paramètres (2D) et à appliquer le résultat vers l'espace 3D. Elle est adaptée au cas des surfaces composées de plusieurs carreaux paramétrés, grâce à une discrétisation préalable des courbes interfaces qui représentent les frontières communes à ces carreaux.

1.2 Données

En entrée, la surface doit être définie par un ensemble de carreaux paramétrés. La description de chaque carreau comprend le domaine de deux paramètres u et v, une fonction analytique $S(u, v)$ de classe C^2, ainsi que les dérivées premières et secondes de $S(u, v)$. Le domaine des paramètres peut être de forme quelconque. Des tests de validation ont été réalisés pour des surfaces sphériques, toriques, de Bézier, B-spline, NURBS, etc. Il est possible de spécifier les tailles souhaitées des éléments (constantes, dépendantes des courbures, ou imposées par des fonctions données).

En sortie, dans la version 0 de ce logiciel, un maillage de surface isotrope est généré. Les versions suivantes offriront en outre des maillages anisotropes, des éléments quadratiques et des maillages adaptatifs (avec des cartes de tailles prescrites sur des maillages de fond).

1.3 Exemples

Les figures ci-après montrent trois exemples de maillages réalisés par le logiciel BLSURF. La figure 1 montre un maillage uniforme et un maillage géométrique de la célèbre théière de l'Utah *(Utah teapot)*, qui est composée de 32 carreaux de Bézier. La figure 2 montre le maillage d'une bouteille de Klein définie par une équation analytique. Ce maillage respecte à la fois une certaine métrique en spirale et la géométrie de la surface.

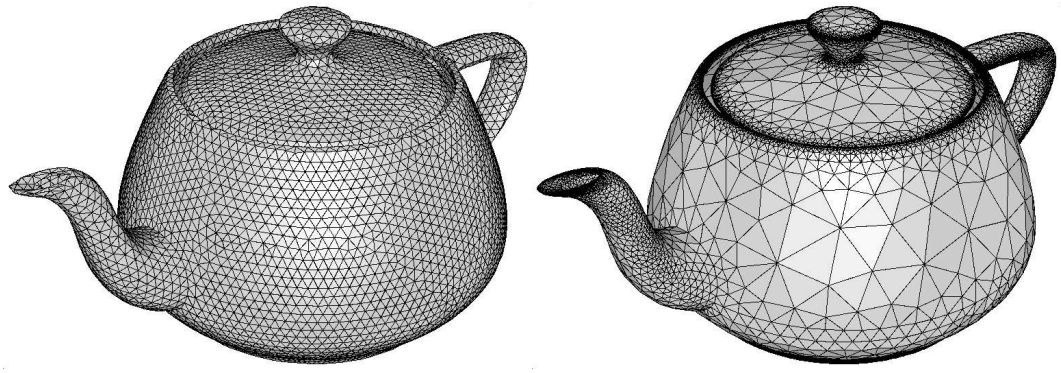


FIG. 1 – *Théière de l'Utah : maillage uniforme et maillage géométrique.*

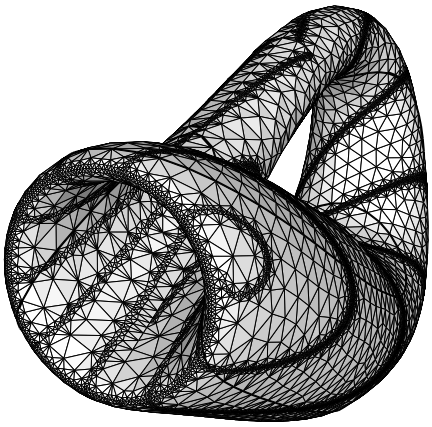


FIG. 2 – *Bouteille de Klein : maillage en spirale et géométrique.*

2 Modélisation d'une surface composée

Cette section donne brièvement le principe de la modélisation d'un seul carreau paramétré, puis d'une surface composée de plusieurs carreaux.

2.1 Carreau paramétré

Considérons un carreau paramétré Σ plongé dans $\mathbb{R}^3$ (voir figure 3 à droite). Par définition, il représente l'image d'un domaine de paramètres σ de $\mathbb{R}^2$ par une fonction $S(u,v)$:

$$\begin{bmatrix} u \\ v \end{bmatrix} \in \sigma \qquad \longmapsto \qquad S(u,v) = \begin{bmatrix} x(u,v) \\ y(u,v) \\ z(u,v) \end{bmatrix} \in \Sigma \ .$$

Le domaine σ contient des segments courbes particuliers, notamment ceux qui forment sa frontière. Il peut aussi contenir des segments courbes internes prédéfinis, par exemple pour modéliser des fissures. Chacun de ces segments courbes γ de $\mathbb{R}^2$ est l'image par une fonction $C(t)$ d'un intervalle $[a,b]$ de $\mathbb{R}$:

$$t \in [a,b] \qquad \longmapsto \qquad C(t) = \begin{bmatrix} u(t) \\ v(t) \end{bmatrix} \in \gamma \ .$$

Connaissant les fonctions $S(u,v)$ et $C(t)$, le segment courbe Γ de $\mathbb{R}^3$ est défini par la fonction $S(u(t),v(t))$, $t \in [a,b]$.

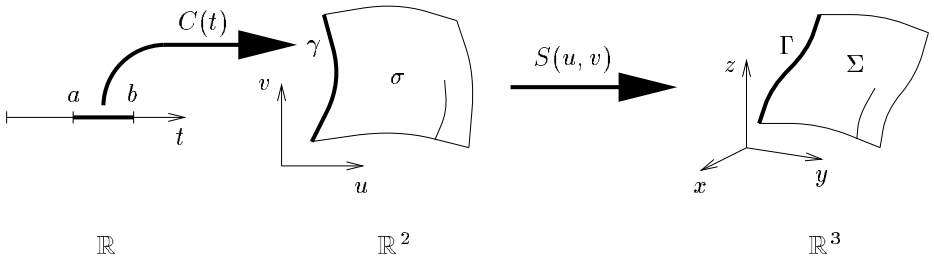


FIG. 3 – *Surface paramétrée simple.*

Nota : dans la suite, pour alléger le texte et par abus de langage, on pourra employer le terme "courbe" à la place de "segment courbe".

Les fonctions précédentes doivent vérifier certaines propriétés :

($\mathcal{P}_0$) Les fonctions $C(t)$ et $S(u,v)$ doivent être définies et continues.

($\mathcal{P}_1$) Leurs dérivées premières $\frac{dC}{dt}(t)$, $\frac{\partial S}{\partial u}(u,v)$ et $\frac{\partial S}{\partial v}(u,v)$ doivent être définies et continues. En outre, le plan tangent à la surface doit être partout défini, ce qui équivaut à : $\forall (u,v)$, $\frac{\partial S}{\partial u}(u,v) \neq 0$ et $\frac{\partial S}{\partial v}(u,v) \neq 0$, et ces deux vecteurs ne sont pas colinéaires.

($\mathcal{P}_2$) Leurs dérivées secondes $\frac{d^2C}{dt^2}(t)$, $\frac{\partial^2 S}{\partial u^2}(u,v)$, $\frac{\partial^2 S}{\partial u\,\partial v}(u,v)$ et $\frac{\partial^2 S}{\partial v^2}(u,v)$ doivent être définies et continues, dans le cas où les tailles souhaitées des éléments du maillage dépendent des rayons de courbures des surfaces et des courbes.

Si la propriété ($\mathcal{P}_1$) n'est pas vérifiée, les métriques associées sont indéfinies en certains points, et le maillage résultant risque d'être de mauvaise qualité. Pour y remédier, il suffit parfois de donner les valeurs des dérivées en un point voisin (solution approchée). Il est également possible de diviser une surface ou une courbe en plusieurs morceaux, ou encore de modifier sa paramétrisation, comme il est précisé dans les exemples ci-dessous.

Exemple 1 : Division d'une courbe

La figure 4 montre la courbe définie par :

$$u(t) = t, \qquad v(t) = |\sin(t)|, \qquad t \in [-\frac{\pi}{2}, +\frac{\pi}{2}] \,.$$

Puisque la dérivée première de la fonction $v(t)$ est discontinue pour $t = 0$, il est possible de diviser cette courbe en deux parties, $t \in [-\frac{\pi}{2}, 0]$ et $t \in [0, +\frac{\pi}{2}]$.

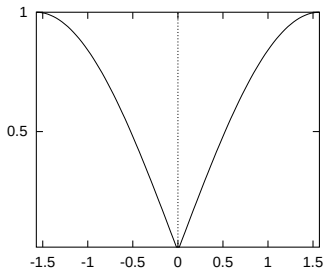


FIG. 4 – *Courbe* $v(t) = |\sin(t)|$.

Exemple 2 : Paramétrisation d'une surface sphérique

Pour paramétrer une surface sphérique, nous cherchons inversement à définir une projection de la surface 3D vers le domaine de paramètres 2D. Sur la figure 5, la projection *orthogonale* du point P de la sphère donne le point P_o, tandis que la projection *stéréographique* donne le point P_s.

Ainsi, la première méthode de projection est dite **orthogonale.** Si $P = (x, y, z)$ est un point de la sphère de centre $C = (0, 0, R)$ et de rayon R, sa projection orthogonale est le point $P_o = (u, v)$ avec $u = x$ and $v = y$. Inversement, un hémisphère peut être paramétré par :

$$S_o(u,v) = \begin{bmatrix} x(u,v) = u \\ y(u,v) = v \\ z(u,v) = \sqrt{R^2 - u^2 - v^2} \end{bmatrix} .$$

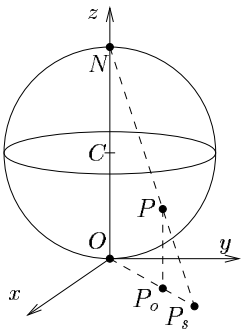


FIG. 5 – *Projections orthogonale et stéréographique.*

L'inconvénient principal de cette paramétrisation est d'être très instable près de "l'équateur", car $\frac{\partial z}{\partial u}(u,v)$ et $\frac{\partial z}{\partial v}(u,v)$ tendent vers l'infini quand $R^2 - u^2 - v^2$ tend vers 0. À titre d'illustration, la figure 6 montre le maillage uniforme d'un disque plan (en bas) et le maillage surfacique correspondant (en haut à gauche).

Une autre méthode de projection, appelée **stéréographique,** est utilisée en cartographie pour les régions polaires. Si $N = (0, 0, 2\,R)$ est le "Pôle Nord" de la sphère, la projection stéréographique du point P est définie comme l'intersection P_s de la droite (NP) avec le plan $z = 0$. Nous avons maintenant $P_s = (u, v)$ avec $u = \frac{2\,R\,x}{2\,R - z}$ et $v = \frac{2\,R\,y}{2\,R - z}$. Inversement, la sphère sans le point N peut être paramétrée par :

$$S_s(u,v) = \frac{2\,R}{u^2 + v^2 + 4\,R^2} \begin{bmatrix} 2\,R\,u \\ 2\,R\,v \\ u^2 + v^2 \end{bmatrix}.$$

Cette dernière paramétrisation est stable près de l'équateur, continûment différentiable, et conserve les angles (mais non les distances) [1]. Par suite, un triangle qui est équilatéral sur le domaine de paramètres le reste sur la sphère, ce qui favorise la qualité des éléments du maillage surfacique (voir figure 6 en haut à droite).

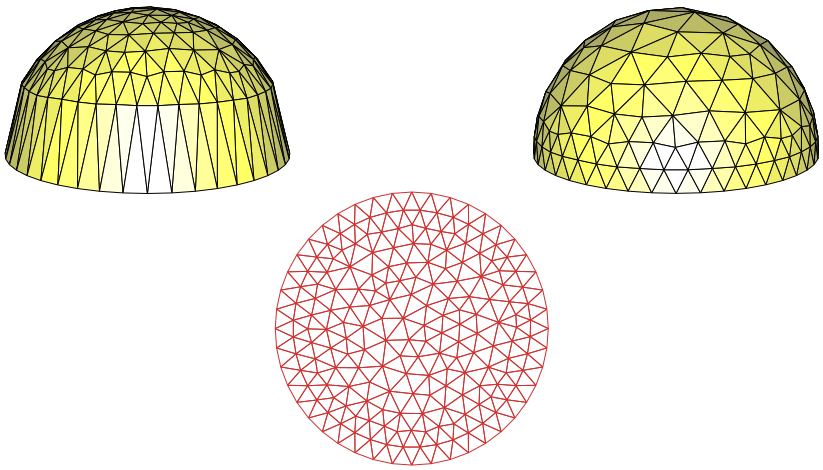


FIG. 6 – *En bas : maillage uniforme d'un disque plan. En haut à gauche : sa projection orthogonale inverse. En haut à droite : sa projection stéréographique inverse.*

Exemple 3 : Paramétrisation d'une surface de révolution

Une surface de révolution peut être définie par une courbe $C(t)$ dans un repère tournant XOY (voir figure 7). Les équations de la courbe et de la surface sont alors respectivement :

$$C(t) = \begin{bmatrix} X(t) \\ Y(t) \end{bmatrix} \quad \text{et} \quad S(\varphi, t) = \begin{bmatrix} X(t)\ \cos\varphi \\ X(t)\ \sin\varphi \\ Y(t) \end{bmatrix} .$$

Pour fixer les idées, prenons pour intervalles de définition $t \in [0,1]$ et $\varphi \in [0, \frac{\pi}{2}]$. En outre, afin de montrer l'influence de la méthode de paramétrisation, supposons que $t = 0 \Rightarrow X(t) = 0$. Ainsi, lorsque φ varie de 0 à $\frac{\pi}{2}$ et $t = 0$, $S(\varphi, t)$ se réduit à un point P de l'axe Oz.

Une première méthode est de choisir directement pour paramètres φ et t, le domaine de ces paramètres formant alors un rectangle. Cependant, l'image inverse du point P par la fonction $S(\varphi, t)$ est dans ce cas un segment entier, le côté d'ordonnée $t = 0$. Pour tous les points (φ, t) de ce côté, la fonction $S(\varphi, t)$ est constante lorsque φ varie, et donc la dérivée $\frac{\partial S}{\partial \varphi}(\varphi, t)$ reste nulle (ce que l'on peut aussi retrouver aisément à partir des équations précédentes).

Une meilleure approche consiste à prendre pour domaine de paramètres le quart de disque dans le repère (u, v) tel que $u = t\ \cos\varphi$ et $v = t\ \sin\varphi$. L'image inverse du point P se réduit alors à un seul point, le centre du disque, ce qui élimine le problème précédent.

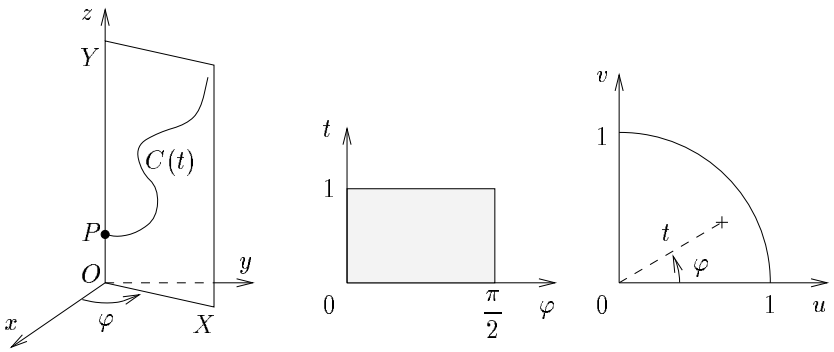


FIG. 7 – *Surface de révolution.*

Après avoir étudié la paramétrisation d'un seul carreau, considérons à présent celle d'une surface composée de plusieurs carreaux.

2.2 Surface composée

Une surface composée est définie par un ensemble de carreaux (voir figure 8). Chaque carreau est défini par un domaine σ_i et une fonction $S_i(u,v)$.

Les images dans $\mathbb{R}^3$ de certaines courbes de $\mathbb{R}^2$ peuvent être géométriquement confondues. Généralement, on souhaite qu'elles soient aussi *topologiquement* confondues, c'est à-dire que le maillage surfacique s'appuie sur une discrétisation unique de ces courbes (appelées *courbes interfaces*). Ceci permet de générer des maillages *conformes* (voir en figure 9 des exemples de maillages conforme et non conforme). Pour cela, on repère les courbes interfaces par des *références* identiques. Dans l'exemple de la figure 8, la courbe tridimensionnelle C_{10} est l'image par $S_1(u,v)$ de la courbe c_{10} du domaine σ_1 et l'image par $S_3(u,v)$ de la courbe c_{10} du domaine σ_3, d'où la référence commune égale à 10.

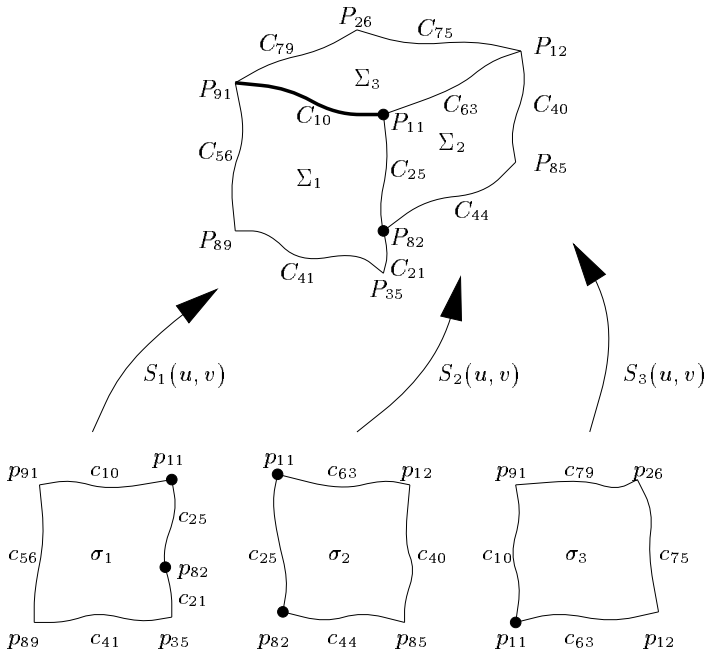


FIG. 8 – *Surface paramétrée composée.*

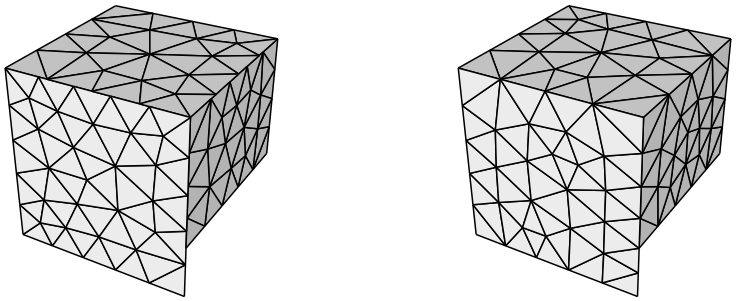


FIG. 9 – *Maillages non conforme (à gauche) et conforme (à droite).*

De même, les points confondus dans $\mathbb{R}^3$ sont repérés par des références identiques. Ici, le point P_{11} est l'image de trois points différents p_{11}. Enfin, pour rendre conformes les carreaux Σ_1 et Σ_2, le point p_{82} doit apparaître dans le domaine σ_1, bien qu'il n'y ait pas de discontinuité entre les courbes c_{21} et c_{25} de ce domaine.

3 Fichiers d'entrée et fonctions externes

En se basant sur la modélisation précédente, le logiciel BLSURF a besoin d'informations sur la paramétrisation des surfaces et des courbes, ainsi que sur les domaines de paramètres et leurs références. À cette fin, ce logiciel peut appeler des fonctions externes regroupées dans un module Fortran de nom `cad_m`, ou encore lire le fichier de données `x.pardom`. En outre, un fichier appelé `blsurf.env` contient des variables d'environnement qui permettent de diriger diverses fonctionnalités du logiciel.

3.1 Module `cad_m` : fonctions externes

Un module Fortran 90 de nom `cad_m` contient l'implémentation de la paramétrisation des surfaces et des courbes (fonctions mathématiques $S(u,v)$ et $C(t)$ ainsi que leurs dérivées). En outre, il contient des fonctions facultatives pour définir la taille souhaitée des éléments. Toutes ces fonctions doivent naturellement être compilées et liées afin de former le fichier exécutable de BLSURF (cf. section 5).

La figure 10 montre la structure générale de ce module. Les fonctions importantes sont `surf0`, `surf1`, `surf2` qui définissent les surfaces, et `curv0`, `curv1`, `curv2` qui définissent les courbes. Les autres fonctions sont facultatives.

```
module cad_m
      ...    ! Cette partie peut contenir des variables globales
contains
      subroutine cad_init
      ...    ! effectue des initialisations, si nécessaire
      subroutine surf0(refs, uv, S)
      ...    ! retourne S(u,v)
      subroutine surf1(refs, uv, Su, Sv)
      ...    ! retourne les dérivées premières de S(u,v)
      subroutine surf2(refs, uv, Suu, Suv, Svv)
      ...    ! retourne les dérivées secondes de S(u,v)
      subroutine curv_int(refs, ic, a, b)
      ...    ! retourne l'intervalle de définition d'une fonction C(t)
      subroutine curv0(refs, ic, t, C)
      ...    ! retourne C(t)
      subroutine curv1(refs, ic, t, Ct)
      ...    ! retourne la dérivée première de C(t)
      subroutine curv2(refs, ic, t, Ctt)
      ...    ! retourne la dérivée seconde de C(t)
      subroutine cad_hphys(refs, uv, h)
      ...    ! retourne la taille souhaitée en un point d'une surface
      subroutine cad_hphyc(refs, ic, t, h)
      ...    ! retourne la taille souhaitée en un point d'une courbe
      subroutine cad_hphyp(refp, h)
      ...    ! retourne la taille souhaitée en un point (extrémité)
      subroutine cad_rads(refs, uv, rhos)
      ...    ! retourne le rayon de courbure d'une surface
      subroutine cad_radc(refs, ic, t, rhoc)
      ...    ! retourne le rayon de courbure d'une courbe
end module cad_m
```

FIG. 10 – *Structure générale du module Fortran 90* `cad_m`.

Nous détaillons ci-dessous chacune des fonctions qui composent ce module (paramétrisation des surfaces et des courbes, puis spécification des tailles). À la fin de ce manuel, un exemple complet de module est donné (cf. section 6.1).

3.1.1 Paramétrisation des surfaces

Rappelons qu'une surface paramétrée simple est définie mathématiquement par une fonction $S(u,v)$. L'implémentation de cette fonction et des ses dérivées est réalisée par les quatre fonctions décrites ci-dessous.

- `cad_init` est appelé une fois pour toutes au début de l'exécution de BLSURF, dans le but d'effectuer certaines initialisations si nécessaire. Il s'agit généralement de la lecture d'un fichier de CAO ou l'équivalent.

- `surf0(refs, uv, S)` implémente une fonction $S(u,v)$. Le paramètre d'entrée `refs` est la référence d'une surface. Le paramètre d'entrée `uv` est un tableau de deux coordonnées ($u =$ `uv(1)` et $v =$ `uv(2)`). En sortie, on obtient le tableau de trois coordonnées `S` $= S(u,v)$.

- `surf1(refs, uv, Su, Sv)` donne, de façon similaire, les dérivées premières `Su` $= \dfrac{\partial S}{\partial u}(u,v)$ et `Sv` $= \dfrac{\partial S}{\partial v}(u,v)$.

- `surf2(refs, uv, Suu, Suv, Svv)` donne, de façon similaire, les dérivées secondes `Suu` $= \dfrac{\partial^2 S}{\partial u^2}(u,v)$, `Suv` $= \dfrac{\partial^2 S}{\partial u \partial v}(u,v)$ et `Svv` $= \dfrac{\partial^2 S}{\partial v^2}(u,v)$. Ces dérivées secondes ne sont réellement nécessaires que dans le cas où les tailles souhaitées des éléments dépendent des rayons de courbures.

La fonction `surf0`, qui implémente $S(u,v)$, peut être programmée intégralement par l'utilisateur, ou faire un simple appel à un système de CAO. Les deux exemples ci-dessous précisent ces deux cas. Une fois terminée la programmation de cette fonction, ses dérivées `surf1` et `surf2` peuvent être écrites manuellement, ou encore obtenues automatiquement par un système de calcul formel tel que Maple [5] ou Odyssée [6].

Exemple 1 : programmation intégrale

La bouteille de Klein représentée sur la figure 2 est définie par les équations paramétriques suivantes :

$$u, v \in [0, 2\,\pi] \qquad\qquad r = 4 - 2\,\cos u$$

$$x = \begin{cases} 6\,\cos u\,(1+\sin u) + r\,\cos u\,\cos v & \text{si } u \in [0, \pi[\\ 6\,\cos u\,(1+\sin u) - r\,\cos v & \text{si } u \in [\pi, 2\,\pi] \end{cases}$$

$$y = \begin{cases} 16\,\sin u + r\,\sin u\,\cos v & \text{si } u \in [0, \pi[\\ 16\,\sin u & \text{si } u \in [\pi, 2\,\pi] \end{cases}$$

$$z = r\,\sin v$$

À cause de la périodicité de cette surface, le domaine des paramètres $[0, 2\,\pi]^2$ est divisé en quatre carrés. Ceci donne la fonction `surf0` suivante :

```
subroutine surf0(refs, uv, S)
  integer :: refs
  double precision :: uv(2), S(3), u, v, cosu, sinu, cosv, sinv, r

  u = uv(1) ; v = uv(2)
  cosu = cos(u) ; sinu = sin(u)
  cosv = cos(v) ; sinv = sin(v)
  r = 4 - 2*cosu
  if (refs <= 2) then
     S(1) = 6*cosu*(1+sinu) + r*cosu*cosv   ! x (cas 1)
     S(2) = 16*sinu + r*sinu*cosv           ! y (cas 1)
  else
     S(1) = 6*cosu*(1+sinu) - r*cosv        ! x (cas 2)
     S(2) = 16*sinu                         ! y (cas 2)
  end if
  S(3) = r*sinv                             ! z
end subroutine surf0
```

Toutes les possibilités du langage Fortran 90 peuvent être mises en œuvre, comme `select case` pour les sélections multiples en fonction de `refs` ou encore l'emploi de tableaux indicés par `refs`.

Exemple 2 : appel à un système de CAO

Le cas d'un appel à un système de CAO est simplement de la forme :

```
subroutine surf0(refs, uv, S)
  integer :: refs
  double precision :: uv(2), S(3)

  call ...  ! appel d'une ou de plusieurs fonctions du systeme de CAO
end subroutine surf0
```

Cette méthode permet en principe de s'adapter à tout système de CAO. En revanche, elle introduit une couche supplémentaire de logiciel, que l'on pourra éliminer pour économiser du temps CPU.

3.1.2 Paramétrisation des courbes

Rappelons qu'une courbe est définie mathématiquement par une fonction $C(t)$, $t \in [a, b]$. L'implémentation de cette fonction et de ses dérivées est réalisée par les quatre fonctions décrites ci-dessous.

- `curv_int(refs, ic, a, b)` donne les bornes de l'intervalle $[a, b]$. Nous verrons que le fichier `x.pardom` contient, pour chaque surface, sa référence `refs` et les descriptions des `nc` courbes qui lui appartiennent (cf. section 3.2). Les paramètres d'entrée de `curv_int`, qui sont `refs` et `ic` ($1 \leq$ `ic` $\leq$ `nc`), désignent une telle courbe.

- `curv0(refs, ic, t, C)` implémente une fonction $C(t)$. Les paramètres d'entrée `refs` et `ic` désignent une courbe comme précédemment. Le troisième paramètre d'entrée `t` est un scalaire compris entre les bornes `a` et `b` retournées par `curv_int`. En sortie, on obtient le tableau de deux coordonnées `C` $= C(t)$.

- `curv1(refs, ic, t, Ct)` donne, de façon similaire, la dérivée première `Ct` $= \dfrac{d}{dt}C(t)$.

- `curv2(refs, ic, t, Ctt)` donne, de façon similaire, la dérivée seconde `Ctt` $= \dfrac{d^2}{dt^2}C(t)$. Cette dérivée seconde n'est réellement nécessaire que dans le cas où les tailles souhaitées des éléments dépendent des rayons de courbures.

Les remarques vues dans la section précédente pour les surfaces (programmation intégrale, appel à un système de CAO et dérivation automatique) s'appliquent également aux courbes.

3.1.3 Spécification des tailles

Dans les cas les plus simples, aucune variable d'environnement n'est modifiée et aucune fonction facultative n'est programmée. On obtient alors un maillage uniforme, la taille prescrite pour les éléments étant $diag/50$, où $diag$ représente la longueur de la diagonale de la boîte englobante (cf. exemples en section 6). Cette taille peut être modifiée par la variable d'environnement `hphydef`.

Dans d'autres cas, le logiciel BLSURF offre de nombreuses possibilités qui sont détaillées ici. Pour obtenir la taille h prescrite en un point P de la surface à mailler, il utilise en fait une taille **physique** h_{phy} (choisie par l'utilisateur) et une taille **géométrique** h_{geo} (destinée à respecter la géométrie de la surface à mailler, en considérant les courbures). Le calcul de ces tailles h_{phy} et h_{geo} est précisée dans les paragraphes (a) et (b) de cette section.

Toute taille physique h_{phy} est bridée, ce qui consiste à la forcer entre deux bornes $h_{phy.min}$ et $h_{phy.max}$]. Plus précisément, si $h_{phy} < h_{phy.min}$ alors $h_{phy} = h_{phy.min}$; sinon si $h_{phy} > h_{phy.max}$ alors $h_{phy} = h_{phy.max}$. De même, toute taille géométrique h_{geo} est bridée entre deux bornes $h_{geo.min}$ et $h_{geo.max}$. En pratique, ces bornes sont données par des variables d'environnement `hphymin`, `hphymax`, `hgeomin` et `hgeomax` (cf. section 3.3). Par défaut, les deux bornes inférieures valent $diag/500$ et les deux bornes supérieures $diag/5$, où $diag$ représente la longueur de la diagonale de la boîte qui englobe la surface à mailler.

Finalement, la taille prescrite h est obtenue de la façon suivante :
- Calculer la taille physique h_{phy} et la *brider*.
- Calculer la taille géométrique h_{geo} et la *brider*.
- La taille prescrite est alors $h = \min(h_{phy}, h_{geo})$.

Il est également possible d'imposer seulement une taille physique tout en ignorant la taille géométrique (ce qui donne finalement $h = h_{phy}$), ou inversement ($h = h_{geo}$).

Précisons à présent le calcul de la taille physique ou géométrique en un point donné P.

(a) Calcul de la taille physique

Ce calcul varie selon l'option choisie, qui est déterminée par la variable d'environnement `hphy_flag`. La valeur de cette variable peut être 0, 1 (par défaut) ou 2.

Si `hphy_flag` $= 0$, la taille physique h_{phy} est ignorée. Dans ce cas, on a obligatoirement `hgeo_flag` $\neq 0$, ce qui donne finalement $h = h_{geo}$.

Si `hphy_flag` $= 1$, la taille est donnée par la valeur de la variable d'environnement `hphydef`.

Si `hphy_flag` $= 2$, elle est obtenue en interrogeant les fonctions `cad_hphys` (surfaces), `cad_hphyc` (courbes) et `cad_hphyp` (points). Chaque fonction peut soit retourner une valeur `h` (qui est ensuite bridée entre les deux bornes `hphymin` et `hphymax`), soit "ne pas répondre" (en n'affectant pas la valeur de `h`), ce qui procure une grande souplesse dans la spécification des tailles. Le calcul diffère selon que le point P est interne à une surface, interne à une courbe, ou à l'extrémité de plusieurs courbes :

- Si le point P est interne à une **surface,** on interroge `cad_hphys`. S'il ne répond pas, on interpole en fonction des valeurs aux sommets des courbes interfaces discrétisées.
- Si le point P est interne à une **courbe,** on interroge d'abord `cad_hphyc`. S'il ne répond pas, on interroge `cad_hphys` pour chacune des surfaces adjacentes et on calcule la moyenne des valeurs retournées. S'il n'y a aucune réponse, on considère les tailles h_1 et h_2 aux deux extrémités de la courbe (voir l'item suivant) et on calcule la valeur interpolée.
- Si le point P est à l'**extrémité** de plusieurs courbes, on interroge d'abord `cad_hphyp`. S'il ne répond pas, on interroge `cad_hphyc` pour chacune des courbes adjacentes et on calcule la moyenne des valeurs retournées. S'il n'y a aucune réponse, on interroge `cad_hphys` pour chacune des surfaces adjacentes et on calcule la moyenne des valeurs retournées. S'il n'y a aucune réponse, on retient la valeur par défaut `hphydef`.

Dans ce qui précède, pour calculer une moyenne de plusieurs valeurs, on utilise par défaut la moyenne arithmétique, mais ceci peut être modifié par la variable d'environnement `hmean_flag`. De même, pour interpoler deux valeurs, on utilise par défaut une interpolation linéaire, mais ceci peut être modifié par `hinterpol_flag` (cf. section 3.3).

Ainsi, dans le cas où `hphy_flag` = 2, les fonctions suivantes doivent être programmées :

- `cad_hphys(refs, uv, h)` retourne (en général) une taille prescrite `h` en un point de la surface `refs` et de coordonnées `uv` dans le domaine des paramètres.
- `cad_hphyc(refs, ic, t, h)` retourne (en général) une taille prescrite `h` en un point de la courbe (`refs`, `ic`) et de paramètre `t`.
- `cad_hphyp(refp, h)` retourne (en général) une taille prescrite `h` en une extrémité de référence `refp`.

Rappelons que chaque programme peut soit retourner une valeur `h` (qui est ensuite bridée entre deux bornes `hphymin` et `hphymax`), soit ne pas répondre (en n'affectant pas la valeur de `h`, cf. exemples en section 6). En effet, l'appel interne au logiciel BLSURF s'effectue de la façon suivante :

```
h = hvide               ! hvide est une constante "tres negative", p.ex. -1.d38
call cad_hphys(..., h)  ! call cad_hphys,   call cad_hphyc   ou   call cad_hphyp
if (h /= hvide) then
   h = ...   ! brider entre hphymin et hphymax
else
   h = ...   ! calculer une autre valeur
end if
```

(b) Calcul de la taille géométrique

Le calcul de la taille géométrique en un point donné P varie selon l'option choisie, qui est déterminée par la variable d'environnement `hgeo_flag`. La valeur de cette variable peut être 0, 1 (par défaut) ou 2.

Si `hgeo_flag` = 0, la taille géométrique h_{geo} est ignorée. Dans ce cas, on a obligatoirement `hphy_flag` $\neq 0$, ce qui donne finalement $h = h_{phy}$.

Si `hgeo_flag` = 1, elle est calculée par le logiciel BLSURF de la façon suivante :
- Si le point P est interne à une **surface,** on calcule $h_s = \lambda\ \rho_s$, où λ est un coefficient et ρ_s est le rayon de courbure de la surface.
- Si le point P est interne à une **courbe,** on calcule la plus petite taille h_s induite par les surfaces adjacentes, et la taille $h_c = \lambda\ \rho_c$, où ρ_c est le rayon de courbure propre à la courbe. Finalement, on retient $h_{geo} = \min(h_s, h_c)$.

– Si le point P est à l'**extrémité** de plusieurs courbes, on calcule la plus petite taille h_s induite par les courbes adjacentes.

Le coefficient λ est calculé de manière à respecter un certain angle de tolérance (défini par la variable d'environnement `angle_mesh`).

Si `hgeo_flag` = 2, les calculs sont similaires mais les rayons de courbures ne sont pas calculés par BLSURF : ils sont donnés par les fonctions externes `cad_rads` pour les surfaces et `cad_radc` pour les courbes. Le but est d'optimiser ces calculs dans certains cas particuliers (sphères ou cylindres par exemple) puisque BLSURF effectue habituellement un calcul plus général.

- `cad_rads(refs, uv, rhos)` retourne le rayon de courbure `rhos` de la surface `refs` au point de coordonnées `uv` dans le domaine des paramètres.
- `cad_radc(refs, ic, t, rhoc)` retourne le rayon de courbure `rhos` de la courbe (`refs`, `ic`) au point de paramètre `t`.

Dans le cas d'un rayon infini (plan ou droite), ces fonctions doivent retourner une "grande valeur", par exemple `huge(1.)` en Fortran 90.

3.2 Fichier d'entrée `x.pardom` : domaines de paramètres

Le fichier `x.pardom` vient compléter la description précédente des différents domaines de paramètres, en donnant essentiellement les références des courbes et des points. Il s'agit à présent d'un simple fichier texte (ASCII) et non d'un programme Fortran.

La forme générale de ce fichier est une structure imbriquée, comme le montre la figure 11. Elle contient un nombre entier `ns`, suivi de `ns` blocs. Chacun de ces blocs contient les entiers `refs`, `orientation`, `nc`, suivi de `nc` blocs, puis l'entier `np` suivi de `np` blocs.

Le détail des différents éléments contenus dans ce fichier est donné ci-dessous. Des exemples complets sont fournis ultérieurement (cf. section 6).

- `ns` : nombre de surfaces (ou carreaux).
- `refs` : référence de la surface.
- `orientation` : entier dont la *valeur absolue* donne un numéro de courbe `ic` et dont le *signe* détermine l'orientation de la surface : si le signe est positif, la surface se trouve à gauche de la courbe `ic` ; s'il est négatif, la surface se trouve à droite. La courbe `ic` est nécessairement de type frontière (`typc` = 1).

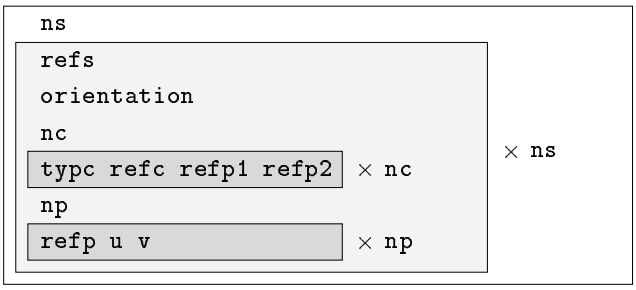


FIG. 11 – *Structure générale du fichier texte* `x.pardom`.

- `nc` : nombre de courbes.
- `typc` : type de la courbe : 1 "frontière", 2 "interne". Une courbe de type 1 fait partie de la frontière d'un sous-domaine. Une courbe de type 2 est interne à un sous-domaine, c'est-à-dire que les arêtes des triangles du maillage final contiennent sa discrétisation. Des exemples de courbes internes sont donnés en section 6.2.
- `refc` : référence de la courbe (qui permet de repérer les courbes identiques dans l'espace 3D).
- `refp1 refp2` : références des extrémités 1 et 2 (obtenues respectivement pour $t = a$ et $t = b$).
- `np` : nombre de points internes.
- `refp` : référence du point interne.
- `u v` : coordonnées du point interne dans le domaine des paramètres.

Remarques

Le nom du fichier est par défaut `x.pardom`, mais son préfixe peut être modifié par la variable d'environnement `pref`.

Si la référence de la toute première courbe est nulle (`is` = `ic` = $1 \Rightarrow$ `refc` = 0), le logiciel BLSURF attribue automatiquement des références en recherchant des points confondus ou voisins (cf. variable d'environnement `eps_glue`). Les résultats sont sauvegardés dans un fichier nommé `x.pardom.new`, qui peut ensuite être réutilisé.

Deux surfaces distinctes ont obligatoirement des références différentes (`is1` $\neq$ `is2` $\Rightarrow$ `refs1` $\neq$ `refs2`). Cette référence est utilisée notamment par la plupart des fonctions du module `cad_m` (cf. section 3.1). Une application commode consiste à demander le maillage d'un seul carreau en donnant `ns` = 1, suivi de la description du carreau contenant sa référence.

Si plusieurs courbes ont une même référence, c'est la définition de la première courbe (par ordre d'apparition dans le fichier `x.pardom`) qui est utilisée par le mailleur.

Pour un carreau donné, les références des frontières doivent être différentes entre elles. Par exemple, pour représenter un cylindre, deux carreaux au moins sont nécessaires à cause de la périodicité de cette surface.

Les références `refp1` et `refp1` des extrémités d'une même courbe `refc` **ne peuvent pas** être identiques. En effet, elles déterminent le sens de parcours de la courbe.

Après avoir lu le fichier `x.pardom`, le logiciel BLSURF s'assure que les propriétés précédentes sont vraies.

3.3 Fichier d'entrée `blsurf.env` : variables d'environnement

Le logiciel BLSURF est gouverné par un ensemble de variables d'environnement. Chaque variable a un nom prédéfini et une valeur par défaut. Le fichier `blsurf.env` permet de modifier les valeurs de ces variables. Il s'agit à nouveau d'un fichier texte (ASCII).

La structure générale de ce fichier est illustrée par la figure 12. Elle permet d'effectuer plusieurs actions successives, qui sont activées par le mot-clé `call`. Le logiciel BLSURF effectue une boucle où il initialise les variables d'environnement avec leurs valeurs par défaut, lit leurs nouvelles valeurs jusqu'au mot-clé `call`, lance l'action correspondante, réinitialise les variables d'environnement, relit les nouvelles valeurs, et ainsi de suite jusqu'à `call exit`. Des exemples complets sont donnés en section 6.

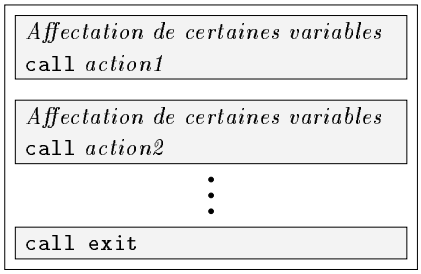


FIG. 12 – *Structure générale du fichier texte* `blsurf.env`.

◊	**Nom**	**Type**	**Valeur par défaut**
	`CheckAdjacentEdges`	entier	0
	`CheckCloseEdges`	entier	0
	`CheckWellDefined`	entier	0
	`CoefRectangle`	réel	0.25
	`LSS`	réel	0.5
	`angle_mesh`	réel	8.
	`angle_smo`	réel	1.
◊	`call`	caractères	`"exit"`
	`element`	caractères	`"p1"`
	`eps_collapse`	réel	0.
	`eps_ends`	réel	$diag/500$
	`eps_glue`	réel	$diag/500$
	`flag`	entier	0
	`format`	caractères	`"mesh"`
◊	`frontal`	entier	1
◊	`hgeo_flag`	entier	0
	`hgeomax`	réel	$diag/5$
	`hgeomin`	réel	$diag/500$
	`hinterpol_flag`	entier	0
	`hmean_flag`	entier	0
◊	`hphy_flag`	entier	1
	`hphydef`	réel	$diag/50$
	`hphymax`	réel	$diag/5$
	`hphymin`	réel	$diag/500$
	`option`	caractères	`""`
	`pref`	caractères	`"x"`
	`refs`	entier	1
◊	`verb`	entier	10

FIG. 13 – *Liste des variables d'environnement du fichier* `blsurf.env`.

Le tableau de la figure 13 donne la liste alphabétique de toutes les variables d'environnement. Celles-ci sont relativement nombreuses, mais seules quelques-unes sont généralement nécessaires en pratique. Elles sont signalées sur la figure par un losange ($\Diamond$). Comme précédemment, *diag* représente la longueur de la diagonale de la boîte qui englobe la surface à mailler. Nous détaillons ci-dessous ces variables en donnant pour chacune d'elles son nom, sa valeur par défaut entre crochets [ ] et sa description.

- `CheckAdjacentEdges` [0], `CheckCloseEdges` [0] et `CheckWellDefined` [0].
  Le logiciel BLSURF appelle n_1 fois le sous-programme `CheckCloseEdges`, n_2 fois `CheckAdjacentEdges` et n_3 fois `CheckWellDefined`. Le but de ces sous-programmes est d'améliorer le maillage des domaines présentant des parties étroites [8] : à chaque itération, `CheckCloseEdges` réduit la taille des arêtes lorsque deux courbes frontières sont proches (voir figure 14), `CheckAdjacentEdges` équilibre les tailles des arêtes adjacentes, et `CheckWellDefined` vérifie que le domaine des paramètres est bien défini. Les trois variables d'environnement ayant les mêmes noms que ces sous-programmes représentent les nombres d'itérations respectifs n_1, n_2 et n_3 (par défaut 0 pour chacun).

- `CoefRectangle` [0.25]. Cette variable définit l'épaisseur relative des rectangles utilisés par le sous-programme `CheckCloseEdges` (voir ci-dessus).

- `LSS` [0.5]. LSS est l'abréviation de "longueur d'un sous-segment". À partir des spécifications de tailles, une métrique Riemannienne $\mathcal{M}$ est définie de manière à ce que les arêtes du maillage désiré soient de longueur 1 dans cette métrique. Pour calculer la longueur $L_{\mathcal{M}}$ d'une arête, on utilise la valeur approchée d'une intégrale. Si la valeur obtenue est inférieure à `LSS`, on considère que le résultat

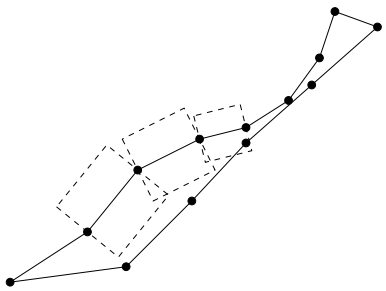


FIG. 14 – `CheckCloseEdges` *: vérification des arêtes proches.*

est correct. Sinon, on subdivise récursivement l'arête en deux sous-arêtes. En guise de conclusion pratique, plus `LSS` est petit et plus les calculs sont précis, mais plus le temps CPU et la mémoire nécessaire augmentent.

- `angle_mesh` [8.]. Cette variable définit un angle θ (en degrés) qui représente la tolérance d'un maillage géométrique, à la fois pour les surfaces et pour les courbes. Dans le cas du maillage d'une surface, θ est l'angle limite entre le plan d'un triangle du maillage et chacun des plans tangents aux trois sommets. Dans le cas de la discrétisation d'une courbe, θ est l'angle limite entre une arête de la discrétisation et la tangente à la courbe. Clairement, plus cet angle est petit et plus le maillage est proche de la surface exacte.

- `angle_smo` [1.]. Cette variable définit un angle θ (en degrés) qui représente la tolérance utilisée pour le support géométrique des courbes (appelé *smooth*). Ce support consiste à approcher les courbes par des segments polygonaux. L'angle θ est l'angle limite entre une arête du segment polygonal et la tangente à la courbe. Pour assurer l'inégalité `angle_smo` $\leq$ `angle_mesh`, BLSURF peut forcer la valeur de `angle_smo`.

- `call` [`"exit"`] active une action prédéfinie de BLSURF, puis redonne aux variables d'environnement leurs valeurs par défaut. Les actions reconnues actuellement sont `"inimesh"` $\Rightarrow$ créer un maillage (qui est le maillage initial dans le cas d'un maillage adaptatif), `"export"` $\Rightarrow$ exporter un maillage dans un fichier, et `"exit"` $\Rightarrow$ sortir du programme.

- `element` [`"p1"`]. Cette variable ne doit pas être modifiée dans la version 0 du logiciel BLSURF. Elle désigne le type des éléments du maillages à créer.

- `eps_collapse` [0.]. Si `eps_collapse` > 0, BLSURF élimine les courbes de longueur inférieure à `eps_collapse`. Ici, pour obtenir une valeur approchée de la longueur d'une courbe, cette dernière est subdivisée arbitrairement en 20 arêtes.

- `eps_ends` [$diag/500$]. Cette variable sert à la détection des courbes de longueur très faible, ce qui constitue parfois une erreur. Un message est imprimé si $\|P_2 - P_1\| <$ `eps_ends`, où P_1 et P_2 sont les extrémités d'une courbe.

- `eps_glue` [$diag/500$]. Sur option, le logiciel BLSURF attribue automatiquement des références en recherchant des points confondus ou voisins (cf. section 3.2). Dans ce cas, deux points P_1 et P_2 sont considérés comme confondus si $\|P_2 - P_1\| <$ `eps_glue`.

- `flag` [0]. Cette variable est utilisée pour modifier le contenu d'un fichier de maillage exporté (cf. section 4). Actuellement, les valeurs possibles sont 0 ou 1.

- `format` ["mesh"]. Cette variable ne doit pas être modifiée dans la version 0 du logiciel BLSURF. Elle désigne le format du fichier à exporter (cf. section 4).

- `frontal` [1]. Si `frontal` = 1, le mailleur insère des points par une méthode frontale. Si `frontal` = 0, il les insère par une méthode algébrique (sur des arêtes internes). Cette dernière méthode est plus rapide mais produit des maillages moins réguliers.

- `hgeo_flag` [0]. Option pour le calcul de la taille géométrique (cf. section 3.1.3). Les valeurs possibles sont 0, 1 ou 2.

- `hgeomax` [$diag/5$]. Taille géométrique maximale (cf. section 3.1.3).

- `hgeomin` [$diag/500$]. Taille géométrique minimale (cf. section 3.1.3).

- `hinterpol_flag` [0] détermine le calcul d'une valeur interpolée v entre deux points P_1 et P_2 sur une courbe. Soit h_1 la valeur au point P_1, h_2 la valeur au point P_2, et un paramètre t qui varie de 0 à 1 en allant de P_1 à P_2. Si `hinterpol_flag` = 0, l'interpolation est linéaire : $v = h_1 + t\ (h_2 - h_1)$. Si `hinterpol_flag` = 1, elle est géométrique : $v = h_1 \left(\frac{h_2}{h_1}\right)^t$. Si `hinterpol_flag` = 2, elle est sinusoïdale : $v = \frac{h1+h2}{2} + \frac{h1-h2}{2} \cos(\pi\ t)$. Voir aussi la section 3.1.3(a) et la figure 23.

- `hmean_flag` [0] détermine le calcul de la moyenne m de n valeurs h_i. Si `hmean_flag` = -1, on obtient le minimum $m = \min_n h_i$. Si `hmean_flag` = 0 ou 2, on obtient la moyenne arithmétique $m = \left(\sum_n h_i\right)/n$. Si `hmean_flag` = 1, on obtient la moyenne géométrique $m = \left(\prod_n h_i\right)^{1/n}$. Voir aussi la section 3.1.3(a) et la figure 22.

- `hphy_flag` [1]. Option pour le calcul de la taille physique (cf. section 3.1.3). Les valeurs possibles sont 0, 1 ou 2.

- `hphydef` [$diag/50$]. Taille physique prédéfinie. Attention, cette taille est bridée dans l'intervalle [`hphymin`, `hphymax`].

- `hphymax` [$diag/5$]. Taille physique maximale (cf. section 3.1.3).

- `hphymin` [$diag/500$]. Taille physique minimale (cf. section 3.1.3).
- `option` [""]. Option utilisée lors de l'exportation des fichiers (cf. section 4). Les valeurs possibles sont "", `"allsurf3d"`, `"allsurf2d"`, `"onesurf3d"`, `"onesurf2d"` et `"smo3d"`.
- `pref` [`"x"`]. Préfixe des fichiers générés par BLSURF.
- `refs` [1]. Référence d'une surface, utilisée lors de l'exportation des fichiers (cf. section 4).
- `verb` [10] définit le pourcentage de "verbosité" du programme. Cette valeur est entière et comprise entre 0 (aucune impression, du moins en théorie) et 100 (maximum d'impressions).

Pour préciser la syntaxe, un fichier d'environnement est formé d'une séquence de couples de mots, où chaque couple représente le **nom** et la **valeur** d'une variable. Il est possible d'inclure des commentaires, compris comme en Fortran 90 entre un point d'exclamation (!) et une fin de ligne. Les mots peuvent être séparés par un ou plusieurs caractères "blancs". Un caractère "blanc" est un espace, une tabulation ou un signe égal (=). Des exemples sont fournis en section 6.

4 Fichiers de sortie (exportation des maillages)

Comme indiqué en section 3.3, l'appel `call export` provoque l'exportation d'un maillage dans un fichier. Généralement, il s'agit du maillage de la surface complète, dans un format assez simple (liste de points et de triangles). Nous donnons ici quelques détails sur cette exportation de maillage.

Lors de l'appel `call export`, le logiciel considère la variable d'environnement `option`.

Si `option = ""` ou `"allsurf3d"`, il exporte le maillage 3D de la surface complète.

Si `option = "allsurf2d"`, il exporte en un seul fichier tous les maillages des domaines de paramètres. Ceci ne peut avoir un intérêt que s'ils ne se superposent pas (voir aussi `"onesurf2d"`).

Si `option = "onesurf3d"`, il exporte le maillage 3D d'un seul carreau, dont la référence est donnée par la variable d'environnement `refs`. Pour obtenir plusieurs fichiers de maillages, il suffit d'appeler plusieurs fois `call export`.

Si `option = "onesurf2d"`, il exporte de la même manière le maillage 2D d'un seul carreau.

Si `option = "smo3d"`, il exporte le support géométrique des courbes (appelé *smooth*).

La variable d'environnement `flag` permet de modifier le contenu du fichier de sortie. Si `flag` = 0, on obtient un maillage surfacique classique comportant une liste de sommets et de triangles. Si `flag` = 1, on obtient en outre la liste des arêtes propres à la discrétisation des courbes (voir figure 17).

Le nom du fichier exporté est par défaut `x.mesh`, mais son préfixe peut être modifié par la variable d'environnement `pref`.

Toutes les fichiers de sortie sont actuellement au format `mesh`, qui est très général (voir l'ouvrage [7], section 10.3, *une base de données générale*). Un tel fichier est formé d'une liste de champs, et nous précisons ci-dessous ceux qui sont utilisés par le logiciel BLSURF.

`MeshVersionFormatted 0` identifie la version du format `mesh`.

`Dimension` est suivi de 2 ou 3 (2D ou 3D).

`Vertices` est suivi du nombre de sommets, puis chaque sommet est de la forme `u v refp` en 2D ou `x y z refp` en 3D. La référence `refp` du point est celle donnée dans le fichier d'entrée `x.pardom`, ou 0 si le point a été généré par le mailleur.

`Triangles` est suivi du nombre de triangles, puis chaque triangle est donné sous la forme `p1 p2 p3 refs` (indices des sommets précédents et référence de la surface).

`Edges` est suivi du nombre d'arêtes d'une courbe discrétisée puis chaque arête est donnée sous la forme `p1 p2 refc` (indices des sommets précédents et référence de la courbe).

`End` indique la fin du fichier.

Pour illustrer les spécifications précédentes, nous donnons ci-dessous quelques exemples de fichiers au format `mesh` (maillages de surfaces 2D et 3D, et discrétisations de courbes 3D).

Surfaces 2D :

```
MeshVersionFormatted 0
Dimension
2
Vertices
9
-2.5000000  -2.5000000   1
 2.5000000  -2.5000000   2
 2.5000000   2.5000000   4
-2.5000000   2.5000000   6
 0.0000000  -2.5000000   0
 2.5000000   0.0000000   0
 0.0000000   2.5000000   0
-2.5000000   0.0000000   0
 0.0000000   0.0000000   1
Triangles
8
7 4 8 1
6 9 5 1
9 8 5 1
1 5 8 1
7 8 9 1
3 7 6 1
6 5 2 1
6 7 9 1
End
```

Surfaces 3D :

```
MeshVersionFormatted 0
Dimension
3
Vertices
9
-2.5000000  -2.5000000   6.2500005   1
 2.5000000  -2.5000000   6.2500000   2
 2.5000000   2.5000000   6.2499995   4
-2.5000000   2.5000000   6.2500000   6
 0.0000000  -2.5000000   3.1250002   0
 2.5000000   0.0000000   3.1249998   0
 0.0000000   2.5000000   3.1249998   0
-2.5000000   0.0000000   3.1250002   0
 0.0000000   0.0000000   0.0000000   1
Triangles
8
7 4 8 1
6 9 5 1
9 8 5 1
1 5 8 1
7 8 9 1
3 7 6 1
6 5 2 1
6 7 9 1
End
```

Courbes 3D :

```
MeshVersionFormatted 0
Dimension
3
Vertices
33
-2.5000000  -2.5000000   6.2500000   1
...
Edges
29
1 2 1
2 3 1
3 4 1
...
End
```

5 Mise en œuvre du logiciel

Le logiciel BLSURF comprend plusieurs fichiers source :

- un programme principal très court, écrit en Fortran, de nom `blsurf.f90`.

```
program blsurf
  use  blsurf_m
  call blsurf_s
end program blsurf
```

- des modules Fortran : `blsurf_m.f90`, `blmc_m.f90`, `share_m.f90` et `sharef_m.f` (ce dernier étant en ancien format fixe, d'où son suffixe `.f`).
- un programme C : `bltms_m.c` (mailleur bidimensionnel).
- un exemple de module `cad_m.f90`.

Généralement, il est nécessaire de récrire le module `cad_m.f90`, qui contient essentiellement l'implémentation de la paramétrisation des surfaces et des courbes. Cette partie peut être programmée intégralement par l'utilisateur, ou faire un simple appel à un système de CAO (cf. section 3.1).

Les fichiers sources précédents doivent être compilés pour créer les fichiers objets correspondants : `blsurf.o`, `blsurf_m.o`, `blmc_m.o`, `share_m.o`, `sharef_m.o`, `bltms_m.o` et `cad_m.o`.

Si le logiciel est autonome, tous ces fichiers objets doivent être liés entre eux (par un éditeur de liens) afin de former le fichier exécutable `blsurf`. L'utilisateur se chargera de créer lui-même les deux fichiers ASCII d'entrée, `x.pardom` et `blsurf.env` (cf. sections 3.2 et 3.3). Il pourra ensuite exploiter les fichiers de sortie (cf. section 4), en utilisant par exemple ses propres outils de visualisation.

Cependant, le logiciel peut aussi être intégré dans un système de CAO existant. Dans ce cas, tous les fichiers objets précédents, à l'exception de `blsurf.o` qui correspond à un programme principal, doivent être liés à ceux qui composent ce système. Ce dernier pourra aussi se charger de générer les fichiers `x.pardom` et `blsurf.env`, ou de relire les fichiers de sortie, de manière transparente pour l'utilisateur. Enfin, les lectures et écritures de fichiers peuvent être remplacées par un passage de paramètres dans le but d'améliorer les performances.

6 Exemples d'utilisation

Dans cette section, nous présentons de façon détaillée l'exemple simple d'un paraboloïde. Ensuite, nous abordons l'exemple célèbre de la théière de l'Utah (32 carreaux de Bézier) puis celui du buste de Victor Hugo (surface discrète cylindrique).

6.1 Paraboloïde

Considérons une partie de paraboloïde définie par :

$$\begin{bmatrix} u \\ v \end{bmatrix} \in [-2.5,\ +2.5]^2 \qquad \longmapsto \qquad S(u,v) = \begin{bmatrix} x(u,v) = u \\ y(u,v) = v \\ z(u,v) = \dfrac{u^2+v^2}{2} \end{bmatrix}.$$

Chaque côté du carré $[-2.5,\ +2.5]^2$ a pour équation $C(t) = P + t\ \overrightarrow{PQ}$, $t \in [0,1]$, où P et Q sont les extrémités du côté. L'implémentation correspondante des surfaces et des courbes est donnée intégralement ci-dessous :

```
module cad_m     ! PARABOLOIDE

  double precision :: segments(4, 4)

contains

  subroutine cad_init
    segments(1,:) = (/ -2.5, -2.5, +2.5, -2.5 /)
    segments(2,:) = (/ +2.5, -2.5, +2.5, +2.5 /)
    segments(3,:) = (/ +2.5, +2.5, -2.5, +2.5 /)
    segments(4,:) = (/ -2.5, +2.5, -2.5, -2.5 /)
  end subroutine cad_init

  subroutine surf0(refs, uv, S)
    integer :: refs
    double precision :: uv(2), S(3), u, v
    u = uv(1) ; v = uv(2)
    S(1) = u
    S(2) = v
    S(3) = (u**2+v**2) / 2
  end subroutine surf0
```

```
subroutine surf1(refs, uv, Su, Sv)
  integer :: refs
  double precision :: uv(2), Su(3), Sv(3), u, v
  u = uv(1) ; v = uv(2)
  Su(1) = 1 ; Su(2) = 0 ; Su(3) = u
  Sv(1) = 0 ; Sv(2) = 1 ; Sv(3) = v
end subroutine surf1

subroutine surf2(refs, uv, Suu, Suv, Svv)
  integer :: refs
  double precision :: uv(2), Suu(3), Suv(3), Svv(3)
  Suu(1) = 0 ; Suu(2) = 0 ; Suu(3) = 1
  Suv(1) = 0 ; Suv(2) = 0 ; Suv(3) = 0
  Svv(1) = 0 ; Svv(2) = 0 ; Svv(3) = 1
end subroutine surf2

subroutine curv_int(refs, ic, a, b)
  integer :: refs, ic
  double precision :: a, b
  a = 0.d0 ; b = 1.d0
end subroutine curv_int

subroutine curv0(refs, ic, t, C)
  integer :: refs, ic
  double precision :: t, C(2)
  C(1) = segments(ic,1) + t * (segments(ic,3) - segments(ic,1))
  C(2) = segments(ic,2) + t * (segments(ic,4) - segments(ic,2))
end subroutine curv0

subroutine curv1(refs, ic, t, Ct)
  integer :: refs, ic
  double precision :: t, Ct(2)
  Ct(1) = segments(ic,3) - segments(ic,1)
  Ct(2) = segments(ic,4) - segments(ic,2)
end subroutine curv1

subroutine curv2(refs, ic, t, Ctt)
  integer :: refs, ic
  double precision :: t, Ctt(2)
  Ctt(1) = 0.d0
  Ctt(2) = 0.d0
end subroutine curv2
```

```
  subroutine cad_hphys(refs, uv, h)
    integer :: refs
    double precision :: uv(2), h
    ! ici, la valeur de h n'est pas affectee
  end subroutine cad_hphys

  subroutine cad_hphyc(refs, ic, t, h)
    integer :: refs, ic
    double precision :: t, h
    ! ici, la valeur de h n'est pas affectee
  end subroutine cad_hphyc

  subroutine cad_hphyp(refp, h)
    integer :: refp
    double precision :: h
    ! ici, la valeur de h n'est pas affectee
  end subroutine cad_hphyp

  subroutine cad_rads(refs, uv, rhos)
    integer :: refs
    double precision :: uv(2), rhos
    print *, "cad_rads should not be called", refs, uv, rhos
    stop
  end subroutine cad_rads

  subroutine cad_radc(refs, ic, t, rhoc)
    integer :: refs, ic
    double precision :: t, rhoc
    print *, "cad_radc should not be called", refs, ic, t, rhoc
    stop
  end subroutine cad_radc

end module cad_m
```

Le fichier `x.pardom`, qui décrit les domaines de paramètres, est donné tout d'abord sans les références des courbes et des points. Ceci a pour effet de générer automatiquement le fichier `x.pardom.new` :

Fichier `x.pardom` :

```
1   ! ns (nombre de surfaces)

1   ! refs (reference de la surface)
1   ! orientation
4   ! nc (nombre de courbes)
1 0 0 0   ! typc refc refp1 refp2
1 0 0 0
1 0 0 0
1 0 0 0
0   ! np (nombre de points internes)
```

Fichier `x.pardom.new` :

```
1   ! ns (number of surfaces)

1   ! refs (reference of the surface)
1   ! orientation
4   ! nc (number of curves)
1 1 1 2   ! typc refc refp1 refp2
1 2 2 4
1 3 4 6
1 4 6 1
0   ! np (number of internal points)
```

Dans un premier temps, le fichier `blsurf.env` appelle les trois fonctions de base de BLSURF et conserve toutes les valeurs par défaut :

Fichier `blsurf.env` :

```
call inimesh
call export
call exit
```

Le fichier exporté a donc pour nom `x.mesh` et contient un maillage uniforme, obtenu par une méthode frontale. La diagonale de la boîte englobante est de longueur $diag = 7.7$, d'où la taille physique par défaut `hphydef` $= diag/50 = 0.154$. La figure 15 représente ce maillage, qui contient 2599 sommets et 4980 triangles.

Un maillage uniforme plus grossier est obtenu en modifiant les fichiers `x.pardom` et `blsurf.env` (voir ci-dessous). Ici, nous imposons un point interne au centre du carré, nous prescrivons une taille `hphydef` $= 1.5$ (en augmentant `hphymax` en conséquence) et nous exportons à la fois le maillage 2D du domaine de paramètres et le maillage 3D de la surface. La figure 16 visualise ces maillages 2D et 3D, qui comportent chacun 45 sommets et 64 triangles.

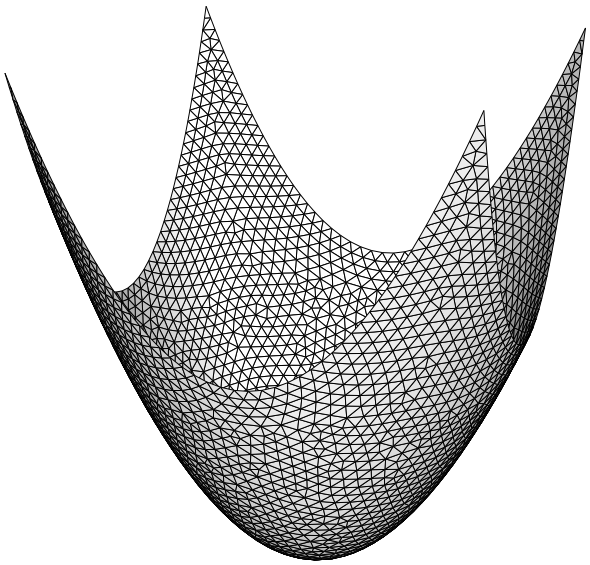


FIG. 15 – *Paraboloïde. Maillage uniforme par défaut.*

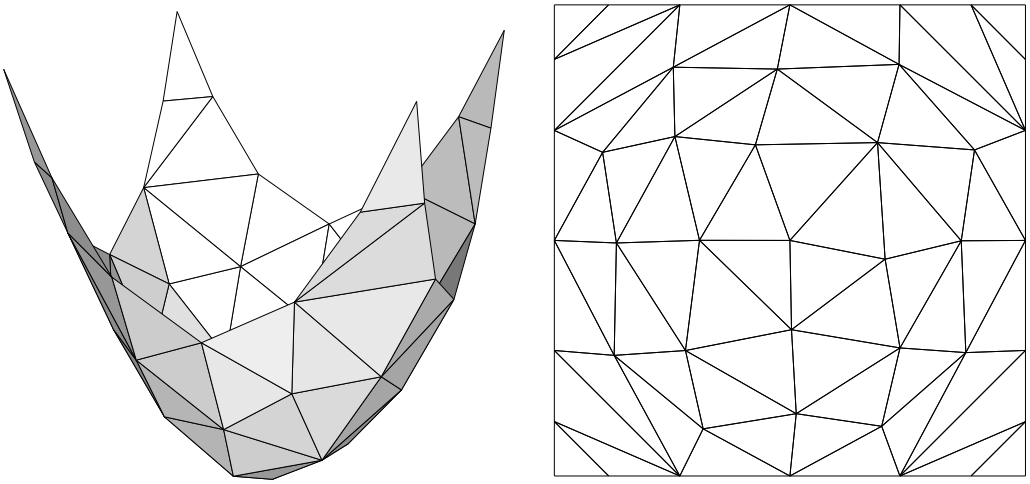


FIG. 16 – *Paraboloïde. À gauche, maillage uniforme grossier avec point central. À droite, maillage du domaine de paramètres correspondant.*

Fichier `x.pardom` :

```
1   ! ns (nombre de surfaces)

1   ! refs (reference de la surface)
1   ! orientation
4   ! nc (nombre de courbes)
1 0 0 0   ! typc refc refp1 refp2
1 0 0 0
1 0 0 0
1 0 0 0
1   ! np (nombre de points internes)
1 0. 0.   ! refp u v
```

Fichier `blsurf.env` :

```
hphydef 1.5
hphymax 1.5
call inimesh
call export

option onesurf2d
pref 2d
call export

option onesurf3d
pref 3d
call export

call exit
```

L'exemple suivant consiste à imposer des lignes internes.

Dans le module `cad_m`, le sous-programme `cad_init` est modifié de façon à spécifier 20 segments de droite et 16 arcs de cercles, et les fonctions `curv0`, `curv1`, `curv2` implémentent la paramétrisation des arcs de cercles :

$$\begin{array}{l} u = u_0 + r\ \cos\theta \\ v = v_0 + r\ \sin\theta \end{array} \quad \text{avec} \quad \theta = \theta_1 + t\ (\theta_2 - \theta_1), \quad t \in [0,1]\ .$$

Les fichiers d'entrée ci-dessous produisent les maillages 2D et 3D de la figure 17. Le choix `flag` = 1 permet de mieux visualiser la discrétisation des courbes frontières et internes. Chaque maillage contient 1772 sommets (+468 extrémités dues à la discrétisation des courbes) et 3366 triangles.

Fichier `x.pardom` :

```
1   ! ns (nombre de surfaces)

1   ! refs (reference de la surface)
1   ! orientation
36  ! nc (nombre de courbes)
1 0 0 0   ! typc refc refp1 refp2
1 0 0 0
1 0 0 0
1 0 0 0
1 0 0 0
1 0 0 0
1 0 0 0
1 0 0 0
2 0 0 0
2 0 0 0
2 0 0 0
...
0   ! np (nombre de points internes)
```

Fichier `blsurf.env` :

```
call inimesh

option allsurf2d
pref 2d
flag 1
call export

option allsurf3d
pref 3d
flag 1
call export

call exit
```

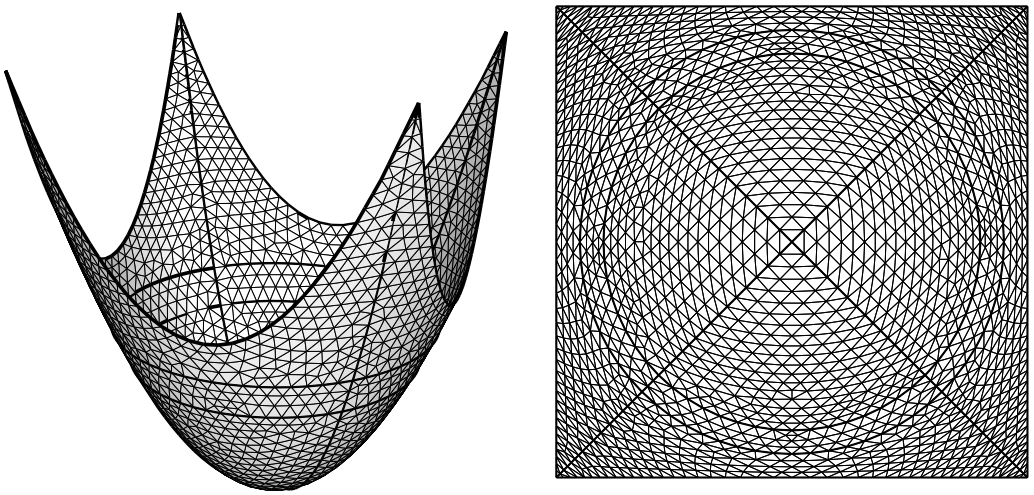


FIG. 17 – *Paraboloïde. À gauche, maillage uniforme par défaut avec courbes internes. À droite, maillage du domaine de paramètres correspondant.*

6.2 Théière de l'Utah

Nous utilisons un exemple célèbre, celui la théière de l'Université de l'Utah *(Utah teapot)* [10]. Elle est composée de 32 carreaux de Bézier : 16 pour le corps, 4 pour la poignée, 4 pour le bec et 8 pour le couvercle (voir figure 18 à gauche).

Un carreau de Bézier cubique est défini sur $[0,1]^2$ par :

$$S(u,v) = \sum_{i=0}^{3} \sum_{j=0}^{3} P_{ij}\ B_i(u)\ B_j(v)\ ,$$

où $P_{ij},\ i = 0..3,\ j = 0..3$ est une grille donnée de points de contrôle, et $B_i(t)$, $i = 0..3$ est la base de Bézier formée des polynômes de Bernstein :

$$\begin{aligned} B_0(t) &= (1-t)^3 \\ B_1(t) &= 3\ t\ (1-t)^2 \\ B_2(t) &= 3\ t^2\ (1-t) \\ B_3(t) &= t^3\ . \end{aligned}$$

Nous avons modifié les frontières du carré $[0,1]^2$ de manière à ce que le maillage du corps, de la poignée et du bec soit totalement conforme. Pour cela, nous avons réalisé indépendamment les maillages surfaciques de différents carreaux, calculé des points aux intersections de ces maillages, et enfin obtenu des splines passant par ces points grâce à l'algorithme de Boor [2].

Pour mieux détailler les exemples, nous présentons des maillages sur les quatre carreaux qui forment l'avant du corps de la théière. La figure 18 à droite montre le maillage uniforme obtenu par défaut. La diagonale de la boîte englobante est de longueur $diag = 4.7$, d'où la taille physique par défaut `hphydef` $= diag/50 = 0.094$. Ce maillage qui contient 1296 sommets et 2430 triangles.

Avec le fichier d'environnement ci-dessous, on exporte les maillages correspondant aux quatre domaines de paramètres de références respectives 1, 4, 5 et 8. Ces maillages sont visualisés en figure 19. On remarque en haut des lignes internes verticales d'abscisses $u = 0.3$ et $u = 0.8$, qui ont été imposées afin de mieux suivre les courbes de niveau sur le rebord supérieur.

Fichier `blsurf.env`

```
call inimesh

option allsurf3d
pref 3d
call export

option onesurf2d
refs 1
pref 2d.1
call export

option onesurf2d
refs 4
pref 2d.4
call export
```

```
option onesurf2d
refs 5
pref 2d.5
call export

option onesurf2d
refs 8
pref 2d.8
call export

call exit
```

En ajoutant simultanément les deux options `hphy_flag 0` et `hgeo_flag 1`, on obtient un maillage géométrique (voir figure 20). Les fichiers de données indiqués donnent respectivement l'angle de tolérance par défaut $\theta = 8°$ ou un angle fixé $\theta = 16°$. La figure 20 montre ces deux maillages : à gauche, 3293 sommets et 6482 triangles ; à droite, 1503 sommets et 2950 triangles. L'étirement de certains triangles est dû à la forte variation locale des rayons de courbures, donc des tailles des éléments (qui sont proportionnelles à ces rayons). Une prochaine version de BLSURF permettra en particulier d'équilibrer ces "chocs" de tailles [3].

L'exemple suivant illustre les spécifications de tailles physiques (voir figure 21). À cette fin, on donne `hphy_flag` = 2 et la fonction `cad_hphys` retourne respectivement les tailles 0.05, 0.1, 0.2 et 0.8 sur les quatre carreaux (de références 1, 4, 5 et 8). Sur les courbes interfaces (hors extrémités), les tailles prescrites sont les moyennes arithmétiques (0.8+0.1)/2 = 0.45, (0.1+0.05)/2 = 0.075, (0.05+0.2)/2 = 0.125 et (0.2+0.8)/2 = 0.5. Au point commun à ces quatre courbes, la taille prescrite est la moyenne arithmétique (0.45+0.075+0.125+0.5)/4 = 0.2875. La figure 21 montre, à gauche, le maillage résultant (475 sommets et 843 triangles). En ajoutant `hgeo_flag` = 1, la géométrie de l'objet intervient (à droite, 3460 sommets et 6784 triangles).

```
subroutine cad_hphys(refs, uv, h)
  integer :: refs
  double precision :: uv(2), h
  double precision, parameter ::    &
    th(8) = (/ 0.05, 0., 0., 0.1,   &
               0.2,  0., 0., 0.8 /)

  h = th(refs)
end subroutine cad_hphys
```

```
subroutine cad_hphyc(refs, ic, t, h)
  integer :: refs, ic
  double precision :: t, h
end subroutine cad_hphyc

subroutine cad_hphyp(refp, h)
  integer :: refp
  double precision :: h
end subroutine cad_hphyp
```

Pour illustrer l'influence du calcul des moyennes, nous reprenons l'exemple précédent (figure 21 à gauche). La variable d'environnement `hmean_flag` a la valeur par défaut 0, ce qui correspond à une moyenne arithmétique des tailles. Avec `hmean_flag` = -1, la taille prescrite sur une courbe est le minimum de celles des surfaces adjacentes, ce qui produit le maillage de la figure 22 à gauche (534 sommets et 960 triangles). Avec `hmean_flag` = 1 (moyenne géométrique), on obtient le maillage de la figure 22 à droite (494 sommets et 881 triangles).

Pour illustrer maintenant l'influence de l'interpolation des tailles prescrites, nous imposons une taille de 0.01 au point central, 0.1 aux extrémités des courbes supérieures, et 0.5 aux extrémités des courbes inférieures. La figure 23 à gauche correspond à l'interpolation linéaire obtenue par défaut (633 sommets et 1188 triangles). La figure 23 à droite correspond à une interpolation géométrique (2748 sommets et 5414 triangles).

```
subroutine cad_hphys(refs, uv, h)
  integer :: refs
  double precision :: uv(2), h
end subroutine cad_hphys




subroutine cad_hphyc(refs, ic, t, h)
  integer :: refs, ic
  double precision :: t, h
end subroutine cad_hphyc
```

```
subroutine cad_hphyp(refp, h)
  integer :: refp
  double precision :: h

  if (refp == 6) then
     h = 0.01
  else if (refp <= 26) then
     h = 0.1
  else
     h = 0.5
  end if
end subroutine cad_hphyp
```

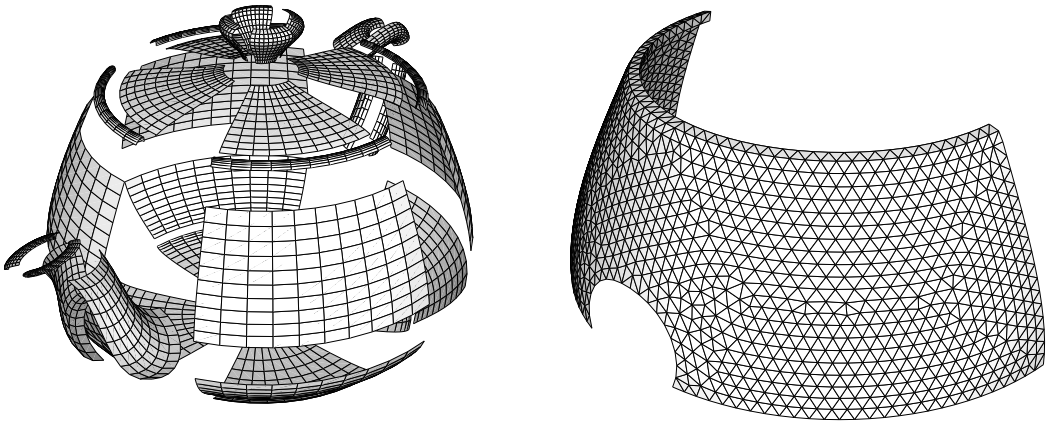


FIG. 18 – *Théière de l'Utah. À gauche, les 32 carreaux de Bézier. À droite, maillage uniforme par défaut de quatre carreaux.*

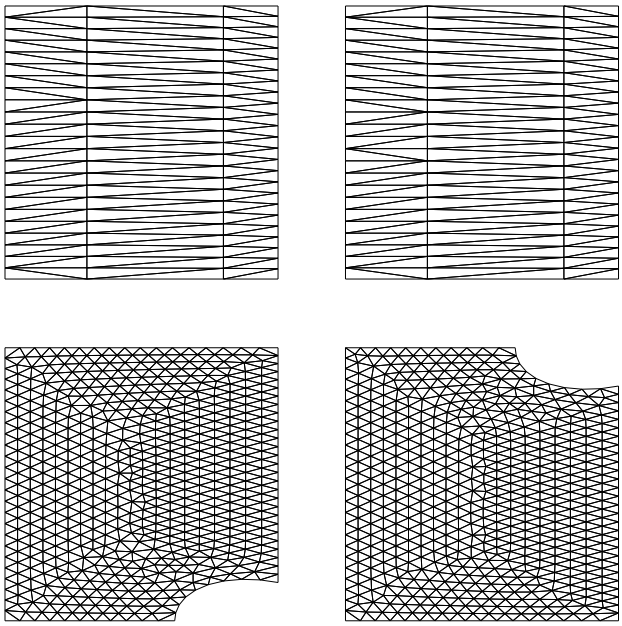
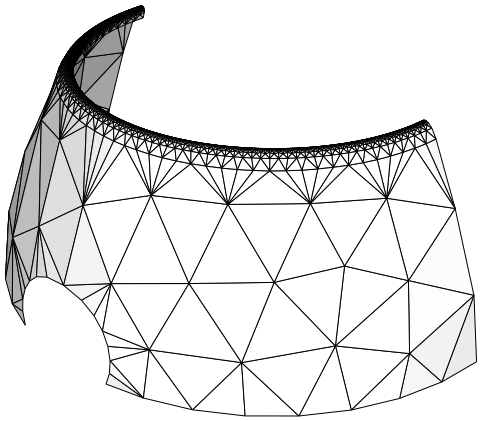
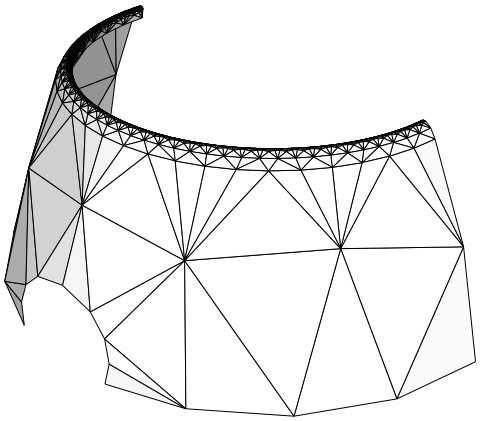


FIG. 19 – *Théière de l'Utah. Maillages des quatre domaines de paramètres correspondant au maillage uniforme précédent.*

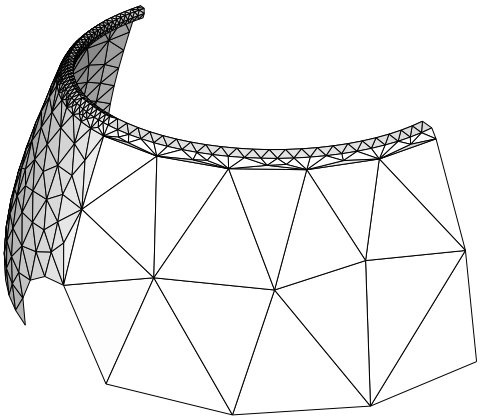


```
hphy_flag 0
hgeo_flag 1
```

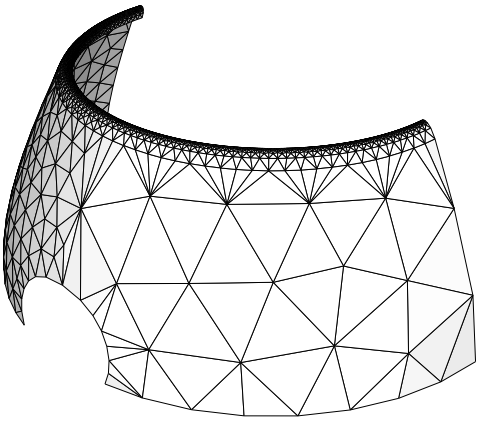


```
hphy_flag 0
hgeo_flag 1
angle_mesh 16
```

FIG. 20 – *Théière de l'Utah. À gauche, maillage géométrique avec un angle par défaut* $\theta = 8°$. *À droite, avec un angle* $\theta = 16°$.

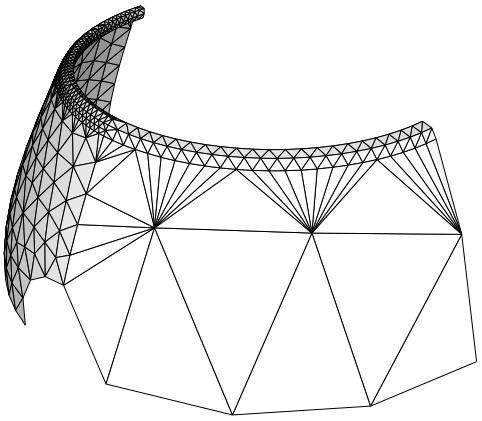


```
hphy_flag 2
```

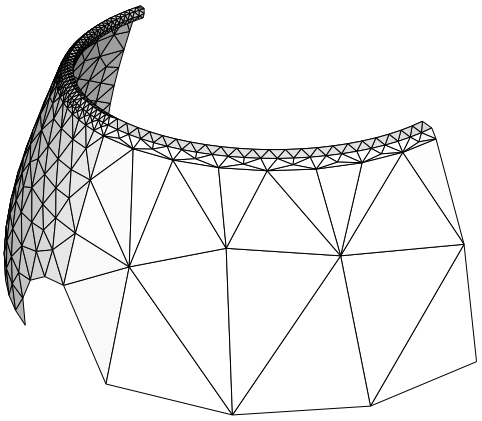


```
hphy_flag 2
hgeo_flag 1
```

FIG. 21 – *Théière de l'Utah. À gauche, tailles constantes par carreau. À droite, mêmes tailles prescrites et prise en compte de la géométrie.*

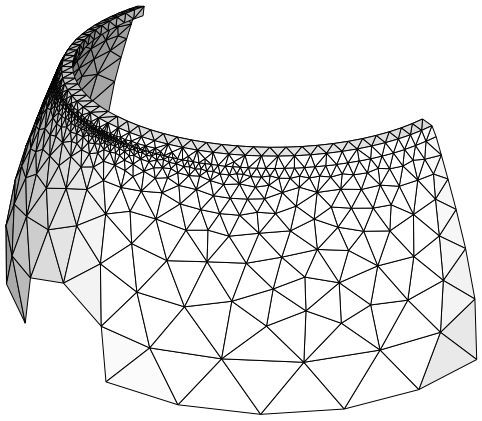


```
hphy_flag 2
hmean_flag -1
```

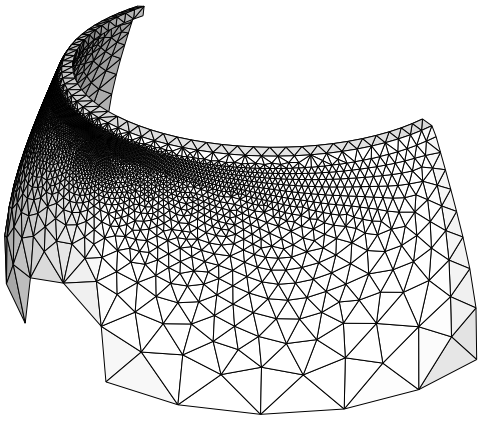


```
hphy_flag 2
hmean_flag 1
```

FIG. 22 – *Théière de l'Utah. Tailles prescrites aux interfaces obtenues par minimum (à gauche) ou par moyenne géométrique (à droite).*



```
hphy_flag 2
hinterpol_flag 0
```



```
hphy_flag 2
hinterpol_flag 1
```

FIG. 23 – *Théière de l'Utah. Interpolation des tailles linéaire (à gauche) ou géométrique (à droite).*

Pour terminer ces exemples, nous montrons en figure 24 le maillage de la théière complète (32 carreaux) avec une taille d'éléments constante $h = 0.1$. Ce maillage contient 5917 sommets et 11658 triangles. À titre indicatif, le temps CPU est de 29 secondes sur une station de travail HP 9000/780, mais ce temps peut être amélioré en optimisant des fonctions externes (`surf0` et `surf1`) ou en modifiant des variables d'environnement (par exemple `LSS` ou `frontal`).

Fichier `blsurf.env`:

```
hphydef 0.1
call inimesh
call export
call exit
```

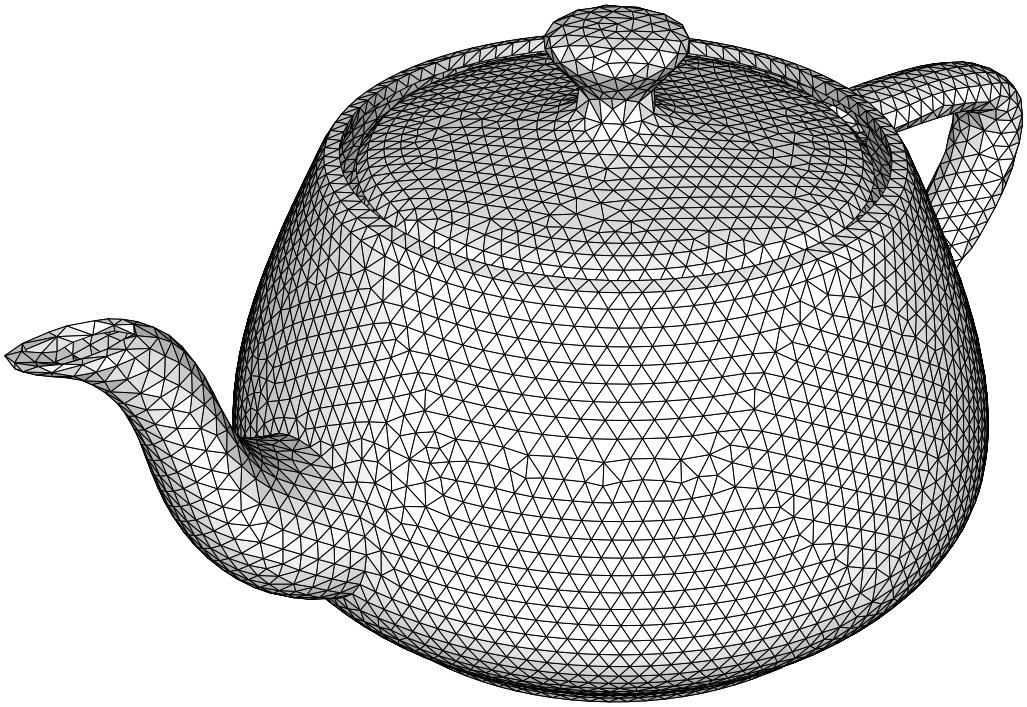


FIG. 24 – *Théière de l'Utah: maillage uniforme.*

6.3 Buste de Victor Hugo

En écrivant les fonctions externes adéquates, le logiciel BLSURF peut être utilisé pour mailler des surfaces définies de manière discrète. À titre d'illustration, nous avons créé un maillage adapté d'une surface discrète cylindrique (le buste de Victor Hugo) obtenue par un procédé de numérisation 3D laser [9] (cf. figure 25).

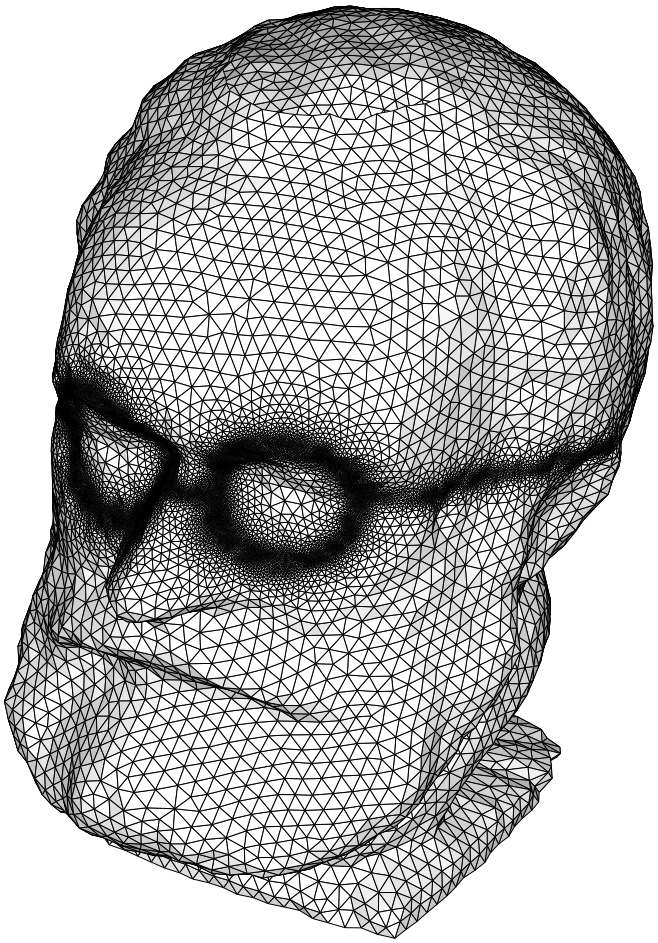


FIG. 25 – *Buste de Victor Hugo (maillage adapté).*

7 Conclusion et extensions futures

La version de BLSURF fournie actuellement présente plusieurs limitations, dont la plupart ont évoquées précédemment. Un certain nombre d'entre elles seront levées dans un court délai :

- Bien que la mémoire soit en général gérée dynamiquement, le maillage de **chaque** carreau est limité à 200 000 points et 400 000 triangles. Si cette limite est dépassée, le message d'erreur *"insufficient memory"* apparaît. Il est alors nécessaire de modifier le fichier source du mailleur bidimensionnel. Ceci sera bientôt remplacé par une gestion plus dynamique de la mémoire.

- Les éléments générés sont de type P1 seulement (triangles à 3 nœuds). La future version génèrera aussi des éléments P2 (triangles à 6 nœuds), Q1 (quadrilatères à 4 nœuds) et Q2 (quadrilatères à 8 nœuds).

- La future version permettra d'équilibrer les "chocs de h" (fortes variations des tailles prescrites) [3].

- La future version permettra de prescrire des tailles à l'aide d'un maillage de fond *(background mesh)*.

- La future version permettra de spécifier une carte anisotrope, c'est-à-dire la taille et la forme des éléments.

- L'utilisation de fichiers auxiliaires (écrits par des composants de BLSURF et relus par d'autres) sera remplacée par des passages de paramètres, ce qui diminue considérablement le temps d'exécution et l'espace-disque.

Malgré ces limitations, il est déjà possible d'intégrer ce logiciel dans un système de CAO, et de réaliser ainsi des maillages surfaciques de bonne qualité dans un temps CPU acceptable.

Références

[1] M. Berger. *Géométrie – tome 2*. Nathan, 1990.

[2] C. de Boor, *A Practical Guide to Splines*, Springer, 1978.

[3] H. Borouchaki, F. Hecht, P. Frey. *Mesh Gradation Control*. International Journal for Numerical Methods in Engineering, vol. 43, n° 6, pp. 1143-1165, 30 November 1998.

[4] H. Borouchaki, P. Laug, P.L. George. *About parametric surface meshing*. 2^{nd} Symposium on Trends in Unstructured Mesh Generation, USNCCM'99, University of Colorado, Boulder, CO, USA, 4-6 août 1999.

[5] B.W. Char et al. *First Leaves: a Tutorial Introduction to MAPLE V*. Springer, 1992.

[6] Ch. Faure, Y. Papegay. *Odyssée User's Guide – Version 1.7*. Rapport Technique INRIA RT-0224, septembre 1998.

[7] P.L. George, H. Borouchaki. *Triangulation de Delaunay et maillage*. Éditions Hermès, Paris, 1997.

[8] P. Laug, H. Borouchaki. *Maillage de l'enveloppe d'une réunion de sphères*. Revue internationale de CFAO et d'informatique graphique, 13(1), pp. 43-64, mars 1998.

[9] F. Schmitt, H. Maître, A. Clainchard and J. Lopez-Krahm. *Acquisition and Representation of Real Object Surface Data*. SPIE Proceedings of Biostereometrics'85 Conference, vol. 602, Cannes, France, 2–6 December, 1985.

[10] A. Watt. *Fundamentals of Three-Dimensional Computer Graphics*. Addison-Wesley, 1989.

Unit´e de recherche INRIA Lorraine, Technopôle de Nancy-Brabois, Campus scientifique,
615 rue du Jardin Botanique, BP 101, 54600 VILLERS LÈS NANCY
Unit´e de recherche INRIA Rennes, Irisa, Campus universitaire de Beaulieu, 35042 RENNES Cedex
Unit´e de recherche INRIA Rhône-Alpes, 655, avenue de l'Europe, 38330 MONTBONNOT ST MARTIN
Unit´e de recherche INRIA Rocquencourt, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex
Unit´e de recherche INRIA Sophia-Antipolis, 2004 route des Lucioles, BP 93, 06902 SOPHIA-ANTIPOLIS Cedex

Éditeur
INRIA, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex (France)
http://www.inria.fr
ISSN 0249-6399